**PROYECTO FINAL DE SEGURIDAD**

**TRANSFERENCIA CIFRADA DE ARCHIVOS**

**Presentado por: Marisol Giraldo y Leonardo Zambrano**

**Presentado al Profesor: Juan Manuel Madrid**

**ETAPA DE ANALISIS**

Este programa permitirá enviar un archivo entre dos computadores, de manera cifrada. Se empleará para el cifrado el algoritmo AES, con clave de 128 bits. La clave por emplear será una clave de sesión, generada utilizando el algoritmo Diffie-Hellman. El programa emisor y el receptor calcularán un hash MD5 del archivo no cifrado, para garantizar la integridad del proceso.

**Recursos**

* Eclipse IDE for Java
* JRE 1.8

**ETAPA DE DISEÑO**

A continuación, se muestra el diagrama de clases de la aplicación de transferencia cifrada de archivos.

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

*Gráfico 1: Diagrama de Clases*

**ETAPA DE IMPLEMENTACIÓN**

**Procedimiento:**

1. Se ejecuta la clase principal **InterfaceEncryptedFilesTransfer** que inicializa la interfaz de la aplicación distribuida de la siguiente manera:

* **SECCIÓN INFORMATIVA DE HOST IP LOCAL Y HOST IP REMOTA**

En esta sección el programa cargará la IP del host local donde se esté ejecutando la aplicación y el usuario podrá ingresar la IP del host remoto del equipo a donde realizará la transferencia de archivo.

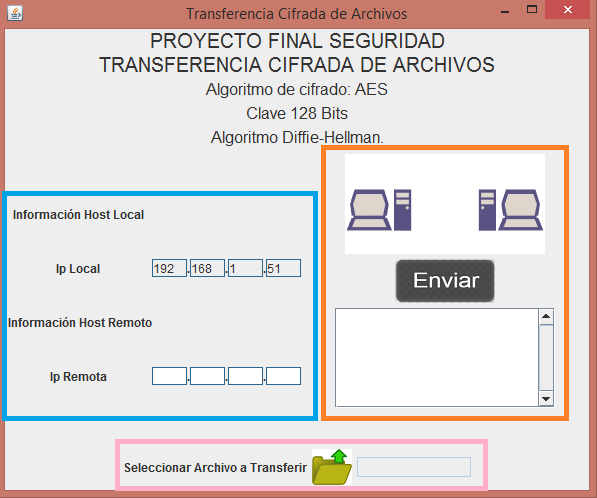
* **SECCIÓN SELECCIONAR ARCHIVO**

En esta sección el usuario podrá seleccionar el archivo que desea transferir.

* **SECCIÓN TRANFERENCIA DE ARCHIVOS**

En esta sección el usuario podrá enviar el archivo al host remoto y recibirá una confirmación de la lista de archivos transferidos con éxito.

A continuación, se pueden visualizar las secciones anteriormente mencionadas:



*Gráfico 2: Interfaz Transferencia Cifrada de Archivos*

1. Se establece la conexión entre el Servidor y el Cliente para realizar la transferencia de archivos.

El paradigma *Cliente/Servidor* se usa para describir un modelo de interacción entre dos procesos, que se ejecutan de forma simultánea, basado en una serie de preguntas y respuestas, que asegura que, si dos aplicaciones intentan comunicarse, una comienza la ejecución y espera indefinidamente que la otra le responda y luego continua con el proceso.

El servidor simplemente espera a la escucha en el *socket* a que un cliente se conecte con una petición. El cliente conoce el *nombre de la máquina* sobre la que está ejecutándose el servidor y el *número de puerto* al que está conectado.

Esta interacción se implementa en las clases **ClientConexion** y **ServerConexion**.

Para la comunicación entre estos se establece una canal público confiable que emplea el protocolo TCP/IP en la clase **ServerTCP** y **ClientTCP**. Una vez se establece la conexión, el servidor y el cliente generan una clave compartida por medio del **Algoritmo Diffie-Hellman.**

1. Se genera una clave compartida entre Servidor y Cliente mediante utilizando el Algoritmo Diffie-Hellman

La criptografía es utilizada para otorgar privacidad, confidencialidad y seguridad. En el envío de información, es importante tener en cuenta el cifrado de mensajes, puesto que evita que terceras personas conozcan el mensaje de tal manera que sea privado entre las fuentes.

El algoritmo AES (Advanced Encryption Standard) es un método criptográfico monoclave, esto quiere decir que se usa la misma clave para cifrar y descifrar, mediante una serie de bucles que se repiten: **10 ciclos para claves de 128 bits**, 12 para 192 y 14 para 256.

En la Clase **ServerTCP**, el algoritmo DH comienza con un número primo grande, **P** y un generador, **G**. Mediante el servicio **SecureRandom** generamos números pseudoaleatorios para el numero **G** de un tamaño de 1024 bits. Una vez generado el numero **G** es enviado al Cliente. En la Clase **ClientTCP** el cliente recibe el numero **G**, y procede mediante el servicio **SecureRandom** a generar el numero **P** de un tamaño de 1024 bits.

Mediante el servicio **KeyParGenerator** se genera pares de claves pública-privada. obteniendo una instancia de esta clase con el método *getInstance("DH")*. Mediante el servicio **KeyAgreement** cliente y servidor coinciden de forma segura en una clave.

Posterior a esta generación, se realiza el intercambio de claves publicas para posteriormente crear la clave privada común.

Mediante el servicio **SecretKeySpec** se genera unaclave de 128 bits adecuada para AES, que tenga al menos 16 bytes. Si el array tiene más, el constructor admite dos parámetros para indicar el byte inicial (0) y cuántos bytes se deben coger (16). El último parámetro es el algoritmo para el que queremos la clave "*AES*".

1. Mediante el servicio de Cipher se realiza la encriptación del archivo para que sea transferido de manera segura.

La clase **MessageDigest** maneja el resumen de los mensajes. Esta clase tiene un constructor protegido por lo que se debe acceder a ella a través del método *getInstace(),* donde el parámetro es el algoritmo que queremos resumir. La salida son 128 bits que codifican 16 caracteres para el MD5.

Se procede a encriptar del archivo a transferir mediante el método ***cipherFile()*** implementado en la Clase **EncryptFil**e.

Con el servicio **Cipher** adecuado para AES con *Cipher.getInstance(“AESECB/PKCS5Padding”).* Se inicializa el cifrador para modo encriptar pasando la clave con *init(Cipher.ENCRYPT\_MODE, key)* y finalmente podemos encriptar el texto con *doFinal(file)* donde file está en Bytes.

El servidor envía el archivo cifrado junto con el HASH MD5 calculado previamente.

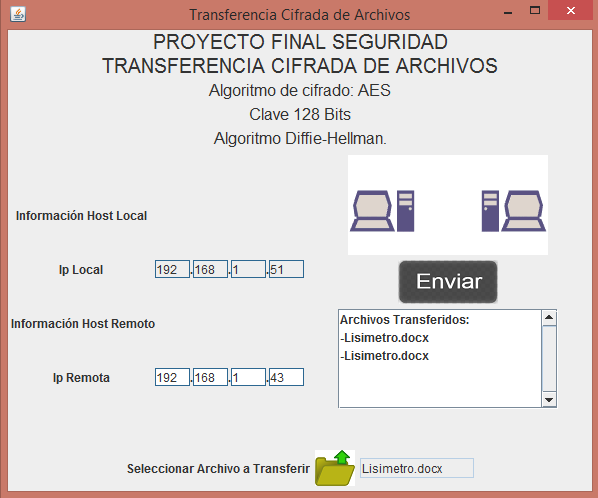
1. Si las claves son correctamente verificadas, mediante el servicio de Cipher se realiza la desencriptación del archivo para que el cliente pueda acceder a él.

El cliente acude a la clase **DecryptFile** para desencriptar el texto, utilizando el método ***decipherFile()*** inicializando el cifrador en modo desencriptar pasándole la clave con *init(Cipher.DECRYPT\_MODE, key)* y se desencripta el texto con *aesCipher.doFinal(encriptedFile)* donde encriptedFile está en Bytes.

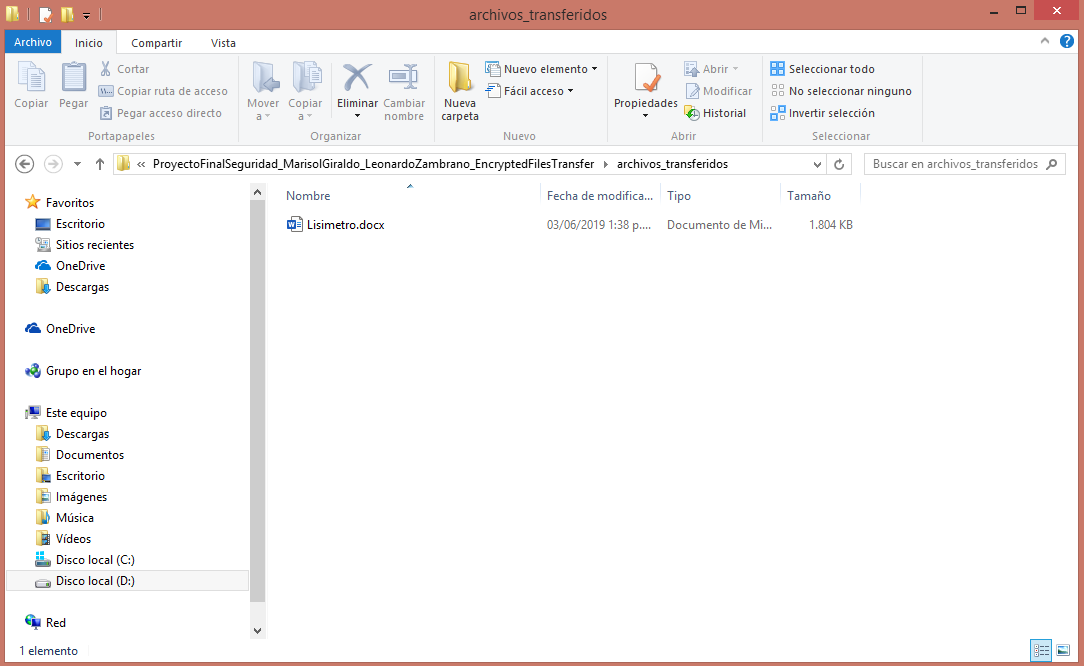
1. El archivo estará disponible en la carpeta archivos\_transferidos

**ETAPA DE PRUEBAS**

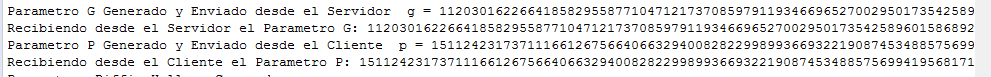
A continuación, se muestra la verificación de que el programa funciona correctamente.



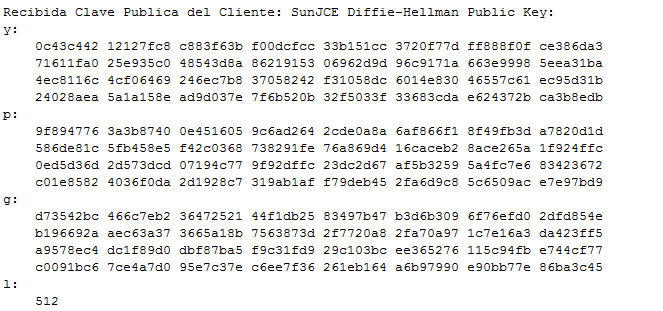
*Gráfico 3: Transferencia de Archivos Exitosa*

**

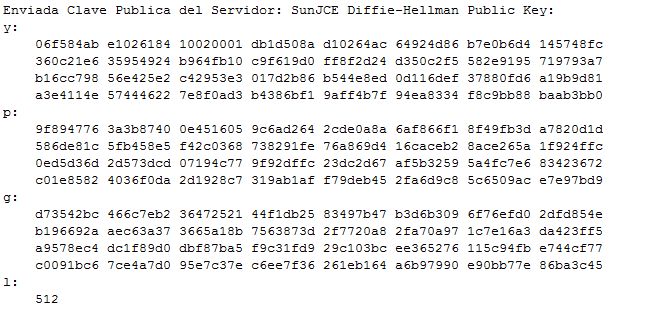
*Gráfico 4: Archivo Transferido a la carpeta archivos\_transferidos*



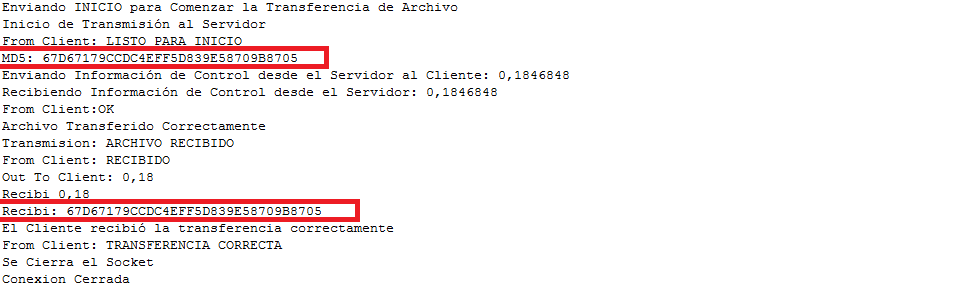
*Gráfico 5: Verificación de Generación y Envío del Parámetro G y P*



*Gráfico 6: Clave Publica del Cliente*



*Gráfico 7: Clave Publica del Servidor*

**

*Gráfico 8: Confirmación MD5 y Transferencia correcta* de archivo

**BIBLIOGRAFIA**

Encriptación y desencriptación en java

<https://ictblog.luisalbertogh.net/?p=274>

Uso de las Extensiones Criptográficas de Java

<https://www.owasp.org/index.php/Uso_de_las_Extensiones_Criptogr%C3%A1ficas_de_Java>

Diffie-Hellman Key Exchange

<https://www.example-code.com/java/dh_key_exchange.asp>

Encriptación con Java

<https://www.scribd.com/document/296731763/Encriptacion-Con-Java>